Conceito Gerais:

→ Definição de Sistema Operativo;
→ Para que são otimizados:
- Em servidores;
- Em *desktop/laptop*;
- Em *tablets/smartphones*;
- Em sistemas embutidos;

→ Breve história do sistema UNIX;
→ Características dos sistemas UNIX;
→ Sistemas mais comuns e a sua linhagem:
- LINUX;
- MAX OS X;
- Windows;
- iOS;
- Android;

→ Funções básicas de um SO;
→ Arquitetura:
- *Kernel*;
- Serviços típicos;
- *System Calls*;

→ Multi-programação e Multi-tarefa;
→ BIOS:
- O que é;
- Para que serve;

→ Bootstrap no UNIX;
→ O que são *interrupts* e *traps* (exemplos);
→ Interação com dispositivos:
- Controladores dos dispositivos;
- *Device drivers*;
- DMA (para dispositivos com elevada largura de banda);

→ Proteção do sistema:
- Modos *kernel* e *user*;
- *System Calls* (como *traps*);
- Implementação;

Sistemas Operativos

Definição do livro: Um sistema operativo é o único programa em execução o tempo todo no computador – geralmente chamado de *kernel*.

Definição da aula: Um sistema operativo é um *software* que providencia os recursos de *hardware* de um computador necessários à execução das aplicações dos seus utilizadores.

- Atua como intermediário entre o utilizador e o *hardware*
- Providencia o básico para as aplicações
- Alguns são desenhados para tornar o sistema do computador conveniente e de fácil utilização, outros são desenhados para usar o *hardware* com maior eficiência e outros para combinar os dois.

Os sistemas operativos são desenhados para corresponder da melhor forma aos dispositivos em que estão.

- **Servidores**: Projetado para maximizar a utilização de recursos e garantir que todo o tempo de CPU, memória e I/O disponíveis sejam usados de maneira eficiente e justa entre todos os utilizadores -***"Keep all users happy!"***
- ***Desktop/Laptop:*** Otimizados para a experiência de um único utilizador em vez dos requisitos de vários utilizadores. Projetado para facilitar a utilização com alguma atenção ao desempenho e quase nenhuma para a utilização dos recursos;
- ***Tablets/Smartphone:*** Otimizados para usabilidade e minimização do gasto da bateria. Projetado para facilidade de uso, com atenção especial à utilização dos recursos;
- ***Embedded Computers:*** A maioria dos computadores embutidos, em dispositivos domésticos e automóveis, têm pouca ou nenhuma interface para o utilizador. Projetado para ser executado sem/mínima intervenção do utilizador;

Um sistema de um computador pode ser dividido em quatro partes:

- ***Hardware*** – recursos básicos do sistema (CPU, memória, dispositivos I/O…);
- **Sistema operativo** – controla e coordena o uso do *hardware* com as várias aplicações e utilizadores;
- **Programas –** definem a maneira como os recursos do sistema são usados para resolver as necessidades dos usuários (processadores de texto, navegadores *web*, sistemas de base de dados, jogos, compiladores, ...)
- **Utilizadores –** pessoas, outros programas/computadores;

O sistema operativo moderno geralmente inclui os seguintes componentes principais:

1) Gestão de **processos**
   - Criar, suspender, retomar e encerrar processos (usuário/sistema);
   - Fornecer mecanismos para comunicação de processos;
   - Fornecer mecanismos para sincronização de processos;
   - Fornecer mecanismo para tratamento de impasses (*deadlock)*
2) Gestão de **memória**;
   - Alocação e desalocação de espaço de memória conforme necessário;
   - Rastreamento de quais partes da memória estão a ser usadas no momento e por quem;

- Decidir quais processos/dados devem ser movidos para dentro e para fora da memória e quando;

3) Gestão de **armazenamento**;
   - Fornecer visão uniforme e lógica do armazenamento de informações;
   - Suporte para criar, excluir e manipular arquivos e diretórios;
   - Políticas de controlo de acesso para determinar quem pode acessar o quê;
   - Mapeamento/*Backup* de arquivos em dispositivos de armazenamento secundário não volátil;
4) Gestão de dispositivos de **I/O**;
   - Oculta peculiaridades dos dispositivos de *hardware* do usuário;
   - Responsável pela gestão de memória de I/O;

O ***kernel*** é o centro essencial de um sistema operativo de computador. É o núcleo que fornece serviços básicos para todas as outras partes do sistema operativo. É a camada principal entre o sistema operativo e o *hardware* e ajuda na gestão de processos e memória, sistemas de arquivos, controle de dispositivos e rede.

Os serviços prestados diferem de um sistema operativo para outro, mas podemos identificar classes comuns:

- Interfaces de usuário – para permitir operação e controle eficazes do sistema **(*User Interface*)**;
- Execução do programa – para carregar um programa na memória e executá-lo **(*Program Execution*)**;
- Operações de I/O – para fornecer um meio de realizar operações de I/O **(*I/O Operations*)**;
- Sistemas de arquivos – para permitir a manipulação eficaz de arquivos e diretórios **(*File Systems*)**;
- Comunicações – para permitir a troca de informações entre processos no mesmo computador ou entre computadores em uma rede **(*Communications*)**;
- Deteção de erros – estar constantemente ciente de possíveis erros que podem ocorrer no *hardware* da CPU/memória, nos dispositivos de I/O ou nos programas do usuário, a fim de tomar as medidas apropriadas para garantir uma computação correta e consistente **(*Error Detection*)**;

Outro conjunto de serviços existe não para ajudar o utilizador, mas sim para garantindo o funcionamento eficiente do próprio sistema:

- Alocação de recursos – quando vários processos estão sendo executados simultaneamente, os recursos disponíveis (como ciclos de CPU, memória principal, armazenamento de arquivos, dispositivos de I/O) devem ser alocados de forma eficiente para cada um deles **(*Resource allocation*)**;
- Contabilidade – para acompanhar quais utilizadores usam quanto e quais tipos de recursos do computador **(*Accounting*)**;
- Proteção e segurança – para evitar que processos simultâneos interfiram uns com os outros ou com o próprio sistema operativo e para proteger o sistema de terceiros **(*Protection and Security*)**;

As ***system calls*** fornecem uma interface para os serviços do sistema operativo. São acessadas a partir de uma *high-level application program interface (API)*. Podem ser agrupadas em seis categorias:

- Controlo de processo (*Process Control);*

- Manipulação de ficheiros (*File Manipulation*);
- Manipulação de dispositivos (*Device Manipulation*);
- Manutenção de informações (*Information Maintenance*);
- Comunicação (*Communication*);
- Proteção (*Protection*);

Um dos aspetos mais importantes dos sistemas operacionais é a capacidade de ter **vários programas em execução.** A **multiprogramação** aumenta a utilização da CPU organizando os processos para que a CPU possa sempre executar um trabalho.

(1) O sistema operativo começa a executar um processo por meio do agendamento de processos (*job scheduling)*;
(2) Eventualmente, o trabalho pode ter que esperar por alguma tarefa, como uma operação de I/O;
(3) Em um sistema não multiprogramado, a CPU ficaria inativa;
(4) Em um sistema multiprogramado, o sistema operativo alterna para outro trabalho. Quando esse trabalho precisa esperar, a CPU muda novamente para outro trabalho e assim por diante. Eventualmente, o primeiro trabalho termina de esperar e recupera a CPU;
(5) Enquanto pelo menos um trabalho precisa ser executado, a CPU nunca fica inativa;

A **multiprogramação** aumenta a utilização da CPU, mas não fornece necessariamente a interação do utilizador com o sistema do computador. ***Multitasking*** (multi-tarefa) é uma extensão da **multiprogramação** que aumenta o tempo de resposta em que a CPU alterna tarefas com tanta frequência que os utilizadores podem interagir com cada tarefa enquanto ela está em execução.

BIOS (Basic Input/Output System) é o primeiro programa responsável pela inicialização do sistema do computador (*bootstrap program)*, é carregada na inicialização ou no *reboot*.

- Armazenada em memória de leitura apenas (ROM, *read-only memory*);
- Inicializa todos os aspetos do sistema, desde dos registros da CPU para controladores de dispositivos até conteúdos da memória;
- Carrega o *kernel* do sistema operativo e inicia a sua execução;

Assim que o *kernel* é carregado, pode começar a fornecer os serviços disponíveis aos utilizadores. Alguns serviços são fornecidos fora do *kernel*, por processos do sistema que são carregados no momento da inicialização (no UNIX, o primeiro processo do sistema é o processo *init* que inicia muitos outros processos do sistema). Assim que esta fase é concluída, o sistema é totalmente inicializado e começa a aguardar a ocorrência de algum evento.

A ocorrência de um evento geralmente é sinalizado por uma interrupção (***interrupt***) do *hardware* ou do *software.* O *hardware* pode acionar uma interrupção a qualquer momento enviando um sinal para a CPU, para comunicar que ele precisa da atenção do sistema operativo. O *software* pode acionar uma interrupção executando uma operação especial chamada de *System Call.*

Quando a CPU é interrompida, suspende a atividade corrente, guardando o estado em que está, e transfere , de imediato, a execução para uma função fixa chamada de *interrupt handler* (manipulador de interrupção) para lidar com o evento. Após a conclusão, a CPU retoma a computação do processo que tinha interrompido.

Traps são levantadas pelo programa do utilizador para invocar uma funcionalidade do sistema operativo. Suponha que o programa do utilizador exija a impressão de algo na tela. Ele invocaria uma ***trap*** e o sistema operativo executaria essa instrução. São usadas principalmente para

implementar ***system calls***. Um ***interrupt*** é gerado por um dispositivo de *hardware*, as interrupções são assíncronas, isto é, podem ocorrer a qualquer momento. Dispositivos como teclados são conectados ao processador através do pino de interrupção. Quando uma tecla é pressionada, ela gera uma interrupção. O processador mudará do processo atualmente em execução para um *interrupt handler*. Nesse cenário, o manipulador de interrupção do teclado é chamado. Depois de completar a rotina de tratamento de interrupção, o processador volta para o programa original que estava sendo executado.

Os sistemas do computador consistem em vários **controladores de dispositivos** (componentes de *hardware*) conectados por meio de um *bus – sistema de comunicação que transfere dados entre componentes dentro de um computador ou entre computadores*. Para se comunicar com cada controlador de dispositivo, os sistemas operativos exigem um ***driver* de dispositivo (*Device drivers*)** específico (componente de software). ***Device drivers*** ajudam o *kernel* a executar ações. São pedaços de código que correspondem a cada dispositivo e são executados quando os dispositivos se conectam ao sistema operativo ou ao *hardware*. Ajudam a fechar o espaço entre as aplicações e o *hardware.* Para garantir a funcionalidade correta, o *kernel* deve ter um ***device driver*** embutido para cada periférico presente no sistema.

Para iniciar uma operação de I/O, o **driver de dispositivo** carrega os registros apropriados no **controlador de dispositivo.**

(1) O controlador, por sua vez, examina o conteúdo desses registros para determinar qual ação tomar (por exemplo, ler um caractere do teclado).
(2) O controlador então inicia a transferência de dados do dispositivo para o *buffer* local.
(3) Uma vez que a operação de I/O foi concluída, o **controlador do dispositivo** informa o ***driver* do dispositivo** por meio de uma **interrupção.** O ***driver* de dispositivo** retorna o controlo ao sistema operativo.

Esta forma de I/O controlada por *interrupt* é boa para transmitir pequenas quantidades de dados, mas pode haver sobrecarga se os dados forem em massa. Para dispositivos de I/O de alta velocidade, capazes de transmitir informações em velocidades próximas às da memória, esse problema é resolvido usando o **acesso direto à memória (DMA – *Direct Memory Access*)**, exemplos, *graphics cards, network cards, disk drive controllers…* O **controlador de dispositivo** transfere blocos de dados de seu próprio armazenamento de *buffer* diretamente para a memória principal, sem intervenção do **driver de dispositivo** (CPU).

Um sistema operativo projetado adequadamente deve garantir que um programa incorreto ou mal-intencionado não possa fazer com que outros programas sejam executados incorretamente. A abordagem adotada pela maioria dos sistemas de computador é fornecer suporte de hardware que nos permita diferenciar entre, pelo menos, dois modos separados de operação:

- ***User Mode;***
- ***Kernel Mode;***

A operação de modo duplo permite que o sistema operativo proteja a si mesmo e a outros componentes do sistema.

Um ***mode bit*** vindo do *hardware* indica o modo atual:

- Permite distinguir quando o sistema está a correr em *user code* ou em *kernel code;*

- Algumas instruções, designadas como instruções privilegiadas, são executáveis apenas no modo kernel (instruções para controle de I/O, gestão do *timer*, gestão de interrupção, …);
- Interrupções ou *system calls* alteram o modo para o *kernel*, o retornar das interrupções ou *system calls* redefinem-no para o modo *user;*

Normalmente, um número é associado a cada *system call* e a *system call interface* mantém uma tabela indexada de acordo com esses números. A *system call interface* invoca a *system call* pretendida no *kernel* do sistema operativo e retorna o status da *system call* e quaisquer valores de retorno.